



**JAVA PROGRAMMING**

(340)

**REGIONAL 2022**

**Production:**

Program 1: Tollway Customer Database (445 points)

**TOTAL POINTS (445 points)**

**Test Time: 90 minutes**

**GENERAL GUIDELINES:**

***Failure to adhere to any of the following rules will result in disqualification:***

1. Member must hand in this test booklet and all printouts if any.
2. No equipment, supplies, or materials other than those specified for this event are allowed in the testing area. No previous BPA tests and/or sample tests (handwritten, photocopied, or keyed) are allowed in the testing area.
3. Electronic devices will be monitored according to ACT standards.

You will have ninety (90) minutes to complete your work.

Your name and/or school name should *not* appear on work you submit for grading.

1. Create a folder on the flash drive provided using your contestant number as the name of the folder.
2. Copy your entire solution/project into this folder.
3. Submit your entire solution/project so that the graders may open your project to review the source code.
4. Ensure that the files required to run your program are present and will execute on the flash drive provided.

\*Note that the flash drive letter may *not* be the same when the program is graded as it was when you created the program.

\*It is recommended that you use relative paths rather than absolute paths to ensure that the program will run regardless of the flash drive letter.

The graders will *not* compile or alter your source code to correct for this.
Submissions that do *not* contain source code will *not* be graded.

**Assumptions to make when taking this assessment:**

* There will only be one record created from the supplied data fields in the comments section.
* Users can attempt to enter in erroneous information during an input prompt.
* The program will only exit at a single point which is when they decline to retrieve the single record created.
* All getter and setter methods for the Customer and Address object have been created; the source code for these classes is not accessible. \
* If the user does not want to update the records the program will ask if they want to retrieve the records. If they answer “no” the program terminates. If they answer yes then the output (supplied in this document) will have NO reference ID.

**Development Standards:**

* Your Code must use a consistent variable naming convention.
* All subroutines (if any), functions (if any), and methods (if any) must be documented with comments explaining the purpose of the method, the input parameters (if any), and the output (if any).
* If you create a class, then you must use Javadoc comments.

***Note to Graders:***

* Output will be static for the customer name and address; however, the data will vary for the monies deposited and the random reference ID generated.
* The output format should be very similar in its organization.
* Error message may differ from sample output.

**Test Case #1.a with Update Entry Error (repeats entry request until Yes or No entered)**

Customer record import successful.

Type in "Yes" if you want **update** this record:

ENTER: Yes or No: ***sssss***

Type in "Yes" if you want update this record:

ENTER: Yes or No:

**Test Case #1.b with Retrieval Entry Error (repeats entry request until Yes or No entered)**

Type in "Yes" if you want update this record:

ENTER: Yes or No: ***no***

Do you want to **retrieve** this record?

ENTER: Yes or No: ***dgsdg***

Do you want to retrieve this record?

ENTER: Yes or No:

**Test Case #2.a Update (“Yes”) with Numerical Entry Error (repeats entry request until double or integer values entered)**

Type in "Yes" if you want update this record:

ENTER: Yes or No: ***yes***

All new records require a new deposit. How much will the customer be depositing?

Please enter in a value between $1.00 to $9,999.99: ***hsfhs***

Please enter a correct value.

Please enter in a value between $1.00 to $9,999.99:

**Test Case #2.b Update (“Yes”) with Numerical Entry Error (program truncates all digits past hundredth place value)**

Type in "Yes" if you want update this record:

ENTER: Yes or No: ***yes***

All new records require a new deposit. How much will the customer be depositing?

Please enter in a value between $1.00 to $9,999.99: ***45.559***

Jose Montana deposited a total of $45.55.

 Reference ID: WVB28039

Do you want to retrieve this record?

ENTER: Yes or No:

**Test Case #2.c Update (“Yes”) with Numerical Entry Error Beyond Parameters (forces required entry)**

Type in "Yes" if you want update this record:

ENTER: Yes or No: ***yes***

All new records require a new deposit. How much will the customer be depositing?

Please enter in a value between $1.00 to $9,999.99: ***33333***

Please enter in a value between $1.00 to $9,999.99: ***.01***

Please enter in a value between $1.00 to $9,999.99: ***345.55***

Jose Montana deposited a total of $345.55.

 Reference ID: QIT44464

Do you want to retrieve this record?

ENTER: Yes or No:

|  |  |
| --- | --- |
| **Solution and Project** |  |
| The project is present on the flash drive |   | 10 points |
| The projects main class is named **TollwayCustomerDataBase** |   | 10 points |
| The class helper method is named **setReferenceID(Customer c)** |   | 10 points |
| The class helper method is named **getDepositMessage(Customer c)** |   | 10 points |
| The class helper method is named **getUserStringInput()** |   | 10 points |
| The class helper method is named **getUserNumberInput()** |   | 10 points |
| The class helper method is named **consoleRecordCheck(Customer c)** |   | 10 points |
| **Program Execution** |  |  |
| The program runs from the USB flash drive |   | 15 points |
| ***If the program does not execute, then the remaining items in this section receive a score of zero.*** |
| The program displays a message declaring successful record import.  |   |  10 points |
| The program prompts and forces user to enter “yes” or “no” if they want to update the record & it is not case sensitive. |   |  10 points |
| If “no” is entered for update: the program prompts and forces user to enter “yes” or “no” if they want to update the record & it is not case sensitive.  |   |  10 points |
| If “yes” is entered for update: the program prompts and forces user to enter “$1.00 to $9,999.99” the deposit amount. All data entry errors are caught. |   |  10 points |
| Program displays customer name and how much was deposited properly formatted: i.e. *Jose Montana deposited a total of $56.00.* |   |  20 points |
| Transaction reference ID is randomly generated with the first three elements being letters and the remaining five are 0 to 9. Note: the letter “O” must be omitted. i.e. *Reference ID: KMU43187*  |   |  20 points |
| The program prompts and forces user to enter “yes: or “no” if they want to retrieve the record & it is not case sensitive. |   |  10 points |
| If “no” is entered for retrieval: the program prompts says “Goodbye” and the terminates.  |  |  10 points |
| If “yes” is entered for retrieval: the program prints the entire record, including the formatted deposit, and the same randomly generated reference ID. |   |  20 points |
| Output matches required format. |   |  20 points |
|  |  |  |
| **Source Code Review** |  |  |
|  |  |  |
| The source code is properly commented |  |  |
|  A comment containing the contestant number is present |   |  10 points |
| Methods and code sections are commented |   |  20 points |
| Code uses try... catch for exception handling for **getUserNumberInput()** when entering numbers.All values entered beyond given range are not accepted; and all values entered into the thousandths decimal place are truncated.  |  |  30 points |
| **getUserNumberInput()**: All values entered beyond given range are not accepted; and all values entered into the thousandths decimal place are truncated. |   |  20 points |
| **main (String args [ ])**: Customer object is correctly constructed from data entry into its given attributes, and also properly passes data into attributes of Address object (attribute in Customer). |  |  10 points |
| **getDepositMessage(Customer c)** method retrieves required fields from Customer and also formats the deposit amount to US currency including “$” and “ ,” plus the cents.  |   | 30 points |
| **setDepositCustomerRecord(Customer c):** calls on *getUserNumberInput()* for data entry and deposits values into customer object; and calls *setReferenceID(Customer c)* to create reference ID. |   |  20 points |
| **setDepositCustomerRecord(Customer c):** calls on *getDepositMessage(c)* for String to print out for feedback to the user.  |   |  10 points |
| **setReferenceID(Customer c):** randomly generates three capital letters, and omits the latter “O”. |   |  20 points |
| **setReferenceID(Customer c):** randomly generates an integer greater than 9,999 and less than 100,000; concatenates with the three letters and returns the value. The reference ID must be in the following format(L: letter & N: number): LLLNNNNNN |   |  20 points |
| **consoleRecordCheck(Customer c):** retrieves all data from the customer object using its getter methods. Places information in proper format |   |  10 points |
| **consoleRecordCheck(Customer c):** formats the deposit retrieved from customer object into US currency including “$” and “ ,” plus the cents. |   |  20 points |
|  |  |  |

**Total Points =\_\_\_\_ / 445 points**

**Suggested Solution**









